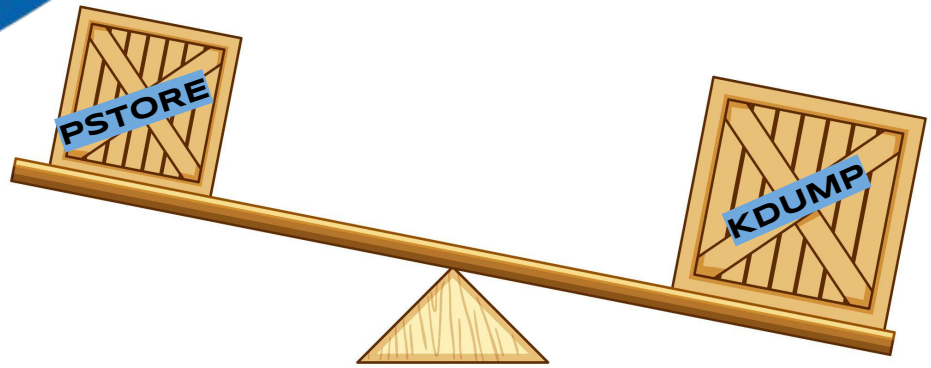

When kdump is way too much

Guilherme G. Piccoli (Igalia)

gpiccoli (at) igalia.com / gpiccoli (IRC)

Linux Plumbers 2023 - Linux Debugging MC

igalia

Talk summary

- Steam Deck game console – Linux based (Arch)
  - Collect logs if panic happens – how/what info?

- Kernel infrastructure for panic data collecting
  - kdump (vmcore but, too “heavy”)

- Alternatives? Pstore!
  - Lightweight, not so much data collected
  - How can we improve this?

The beginning: Steam Deck

- CPU/APU AMD Zen 2 (custom), 4-cores/8-threads
- 16 GB of RAM / 7" display
- 3 models of NVMe storage (64G, 256G, 512G)

SteamOS

- SteamOS version 3 is based on Arch Linux (some extra pkgs on top)
  - Game mode (Gamescope), Desktop mode (KDE Plasma)
  - Gaming layers: Proton (Wine) / DXVK / VKD3D

- What if such complex SW stack crashes?
  - Interest in having log collecting on errors
  - Kernel panics – what action do we take?

- Arch Linux has no official kdump tool
  - Comprehensive wiki, but no automatic tooling

But what should we collect?

- In case of a panic, we could try collecting:
  - vmcore
  - dmesg
  - extra information from userland processes

- But is vmcore too much? Hard to share, storage concerns
  - Is dmesg (call trace) enough? Maybe with some extra info
  - Statistics - logs from lots of users are helpful

- Rely on in-kernel infrastructure for that
  - What tools do we have available?

The good ol' kdump

- Kexec-based solution, new kernel collects data from the broken one
  - As soon the panic happens, jump to a fresh kernel...
  - ...that was preloaded in a reserved / untouched memory region...
  - ...so this new kernel can collect the vmcore of the broken one

- Such vmcore is (usually) compressed and stripped
  - *Post-mortem* analysis: can be inspected later
  - Also shared with others (like support teams)
  - Rich data collection, standard on servers

But not always suitable...

- Pre-reserved memory required (`crashkernel=`)
  - >200M lately, for most distros
  - Reserved on boot, can't adjust without reboot
  - Difficult to estimate properly

- Size of vmcore - could be even in the GB order
  - Privacy: bunch of kernel data ready to be inspected (for good and bad)

- Risks during
  - Crash kernel booting (PCI devices, potential nightmare)
  - vmcore collecting (OOM, makedumpfile bugs, version incompat.)

Alternatives?

- Hypervisor-aided mechanisms
  - fadump (ppc)
  - hv_kmsg (hyper-v)
  - qemu `dump-guest-memory`
- netconsole
  - Or even serial console dump
- **pstore (persistent storage)**
  - Panic time data collection
  - Very flexible - multiple backends

Pstore: the lightweight way

- Saves the kernel log in a persistent storage (backend)
  - Multiple backends: RAM, UEFI, ACPI ERST, block device
- Common in embedded devices - also in chromebooks
  - And the Steam Deck - stay tuned!
- Fast and (hopefully) transparent process
  - Bonus points: no kexec support is required!

Pstore: Pros / Cons

- Various frontends as well: ftrace, console, pmsg (userspace)
- Doesn't require memory reservation - see the UEFI backend!
- Can't collect a full vmcore
- Run after panic notifiers (for now!)
- No tooling (AFAIK) to configure pstore and deal with logs
  - W.r.t logs, we have some elementary tool: systemd-pstore

Presenting: kdumpst

- kdumpst is a new Arch Linux kdump and pstore tool
  - Available on AUR , supports GRUB and initcpio / dracut
  - Includes kdump vmcore collection and sysctl customizations

- Defaults to pstore; currently only ramoops backend
  - Supporting UEFI and systemd-boot planned

- Used by default on Steam Deck, able to submit logs to Valve
  - But how to improve the amount of logs collected?

`panic_print` FTW

- Sysctl/parameter that enables printing extra stuff to dmesg during panic
  - All tasks' status
  - Memory info, CPUs backtraces
  - Lock / Timer info

- May dump too much lines
  - Printing on panic "usual" risks

- Run after panic notifiers (this thing, again!)

Interlude: panic notifiers

- Notifier callbacks: list of functions to be executed in any order
  - Multiple types: atomic callbacks, blocking callbacks, etc
  - Panic notifiers == list of atomic callbacks executed on panic

- Any driver (even OOT) can register a notifier, to do...anything!
  - Risky for kdump reliability
  - But panic notifiers are sometimes necessary
  - "Solution": new kernel parameter (ofc), `crash_kexec_post_notifiers`
  - All-or-nothing option, runs **all** notifiers before kdump

- Refactor proposed, more details in this Kernel Recipes presentation

Challenges/Ideas/Discussion

- Is pstore risky? Panic time data collection
  - Variable risk, depends on the backend

- ramoops limitations - not so easy to reserve some bits of memory
  - Risk of FW corrupting/retraining memory on boot
  - Idea: implement a test for all backends
  - Another idea: a kernel parameter to reserve some ramoops memory

- The panic notifiers risks (addressed on refactor?)

- Too few data (even with `panic_print`)?
  - What else could we collect on panic time?

igalia

Credits: Title Image by brgfx on Freepik